NASA/TM-2009-215764

Baseline Assessment and Prioritization Framework for IVHM Integrity Assurance Enabling Capabilities

Eric G. Cooper and Benedetto L. Di Vito
Langley Research Center, Hampton, Virginia

Stephen A. Jacklin
Ames Research Center, Moffett Field, California

Paul S. Miner
Langley Research Center, Hampton, Virginia

June 2009

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, and organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at 443-757-5803

- Phone the NASA STI Help Desk at 443-757-5802

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2009-215764

Baseline Assessment and Prioritization Framework for IVHM Integrity Assurance Enabling Capabilities

Eric G. Cooper and Benedetto L. Di Vito
Langley Research Center, Hampton, Virginia

Stephen A. Jacklin
Ames Research Center, Moffett Field, California

Paul S. Miner
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

June 2009

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Baseline Assessment and Prioritization Framework for IVHM Integrity Assurance Enabling Capabilities

Eric Cooper, Ben Di Vito, Stephen Jacklin, and Paul Miner

Abstract - *Fundamental to vehicle health management is the deployment of systems incorporating advanced technologies for predicting and detecting anomalous conditions in highly complex and integrated environments. Integrated structural integrity health monitoring, statistical algorithms for detection, estimation, prediction, and fusion, and diagnosis supporting adaptive control are examples of advanced technologies that present considerable verification and validation challenges. These systems necessitate interactions between physical and software-based systems that are highly networked with sensing and actuation subsystems, and incorporate technologies that are, in many respects, different from those employed in civil aviation today. A formidable barrier to deploying these advanced technologies in civil aviation is the lack of enabling verification and validation tools, methods, and technologies. The development of new verification and validation capabilities will not only enable the fielding of advanced vehicle health management systems, but will also provide new assurance capabilities for verification and validation of current generation aviation software which has been implicated in anomalous in-flight behavior. This paper describes the research focused on enabling capabilities for verification and validation underway within NASA's Integrated Vehicle Health Management project, discusses the state of the art of these capabilities, and includes a framework for prioritizing activities.*

Introduction

NASA's Integrated Vehicle Health Management project (IVHM) [1] is incorporating advanced technology for assessing vehicle health at the system and subsystem level. For example, airframe health management will perform in-flight diagnosis and assessment through the integration of sensors, sensory materials, and advanced algorithms for reconstructing damage fields and estimating structural durability and remaining useful life. Many of these algorithms will incorporate advanced information processing technologies including neural networks, expert systems, fuzzy logic systems, pattern recognition, signal processing for spectral analysis and feature extraction, and statistical algorithms for detection, estimation, prediction, and fusion [2]. Propulsion health management systems will incorporate sensor suites and advanced fault diagnosis and prognosis information processing technologies that will assess remaining useful life for use by digital engine controllers and maintenance crews. Aircraft systems health management will utilize advanced information processing technologies for detecting and classifying fault conditions, estimating remaining useful life, developing Bayesian sensor fusion tools, and investigating data-driven and statistical life estimation models. A commonality across many of these health management approaches is the application of large, complex software systems that use algorithms that are dynamic and non-deterministic in nature. Fielded systems employing such algorithms will, in certain cases,

learn and change over time. This class of software presents a certification challenge that is not addressed by current civil aviation airborne software standards such as DO-178B [3, 4]. Moreover many of these health management capabilities will be used in flight or mission-critical functions such as adaptive control and condition-based maintenance whereby routine scheduled-based maintenance inspections and procedures are replaced by on-board automated health assessment systems. The increased reliance on such advanced algorithms in flight or mission-critical applications poses significant verification and validation challenges.

NASA's IVHM project is addressing these challenges by focusing on the investigation, development, and demonstration of tools, techniques, and methodologies that will provide enabling capabilities for assuring design and processing integrity of flight and mission-critical systems. Research is underway in compositional verification, hybrid systems analysis, and a new IVHM concept referred to as software health management. Compositional verification embodies a suite of techniques and approaches that provide strong guarantees that the verification of a system or sub-system can be efficiently accomplished by taking advantage of the verification of individual components. Software health management seeks to develop the tools and techniques needed to enable the detection, diagnosis, prognosis, and mitigation of hazards due to software faults. Hybrid systems analysis extends model checking, theorem proving, and static analysis tools with additional capability for formal analysis of systems comprised of both continuous and discrete behavior.

Finally, it is certainly beyond the scope of NASA's IVHM project to address all of the anticipated verification and validation challenges necessary for deploying advanced IVHM technologies. We propose a framework for prioritizing research activities that takes into account relevance to IVHM technology, relevance to system-level dependability cases, and relevance to broader aviation safety verification and validation needs. The contribution of current research efforts relative to this framework is discussed.

Compositional Verification

The principle behind compositional verification is to use a divide-and-conquer strategy to explore the state space of a large model of an aircraft or aircraft component. Models may be of the form of finite state machines, formal logic representations, or mathematical equations describing the behavior of the system in different modes of operation. Using the compositional verification approach, large and complex system-level models are decomposed into a set of smaller models, which can be verified separately. The complete model is then verified by checking that the assumptions about the environment (i.e., the other small models) hold at the system level.

An advantage of compositional verification is that individual models can be verified using different techniques. In some cases, it may be possible to use model-checking techniques for discrete finite-state machines. Hybrid model-checking techniques can be

used on IVHM models with discrete and continuous components (e.g., to describe the different modes of a complex control system).

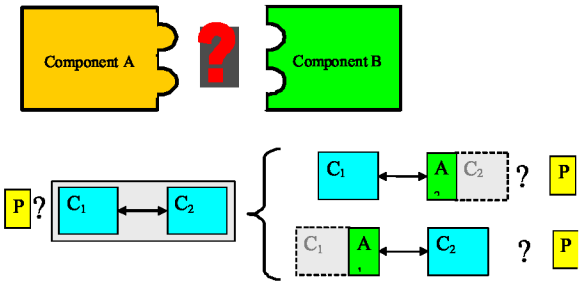

Figure 1: The verification that a property (P) holds for a system can be done through compositional verification by testing that the property holds for (verified) individual system components under assumptions (A) made about the state of the other components.

State-of-the-art in Compositional Verification

Compositional verification offers a means to implement formal analysis tools on industrial size systems that ordinarily would be mathematically intractable using monolithic approaches. Scalability of formal methods to handle large systems is based on a "divide-and-conquer" approach that breaks up the verification of a large system into smaller tasks that involve the verification of its components. Of course, the verification of the smaller pieces is not useful unless there is a means to also verify the correct interaction of the components. This problem is addressed through the application of assume-guarantee reasoning [5, 6]. This technique uses assumptions when checking individual components of a system that essentially encode the expectations that each component has for the rest the system in order to operate correctly.

Derivation of the correct assumptions is a non-trivial manual process and this problem can limit the scope of the compositional verification method. Over the last few years, the Reliable System Engineering group at Ames Research Center has developed a collection of techniques and a supporting toolset for performing assume-guarantee reasoning of software in an automated fashion, which significantly increases the impact of assume-guarantee reasoning in practice. These methods are applicable both at the level of design models, and at the level of actual source code [7-11].

The techniques for compositional verification have been applied to the design and implementation of the Executive of the Ames K9 Mars Rover and to verify safety properties of OpenSSL, an Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols (www.openssl.com). In each case, more than an order of magnitude reduction in the state space was achieved using compositional verification analysis. Further research is needed to examine the expansion of this approach to industrial size IVHM applications, but so far the results look very promising.

Compositional Verification and Adverse Events

An IVHM diagnosis system can be as simple as checking that the value of a physical sensor is in nominal range or as complicated as feeding physical and virtual sensor information to various diagnosis engines and somehow fusing the results in a coherent diagnosis. The current trend suggests that diagnosis engines will become increasingly sophisticated, incorporating advanced technologies such as model-based diagnosis or neural networks. The trend is to keep them quite modular so that a new promising technique can be added as the system changes. Compositional verification is very well suited to perform verification of modularly designed software systems (as explained previously above).

Compositional Verification Baseline Datasets

Although no data has been created by this research effort thus far under NASA's IVHM project, past research has indicated at least an order of magnitude reduction in the state space using compositional verification on problems of moderate size. Using design assumptions for model checking the K9 Rover code with the JavaPathfinder model checker, a ten-fold state space reduction was achieved as compared to the standard monolithic approach summarized in table 1. Equally significant reductions in memory usage and analysis time were realized at the same time.

System	States	Transitions	Memory	Time
Monolithic	183,132	425,641	952.85Mb	12m,24s
Premise 1	53,215	117,756	255.96Mb	4m,49s
Premise 2	13,884	20,601	118.97Mb	1m,16s

Table 1: State-space reduction using model checking design assumptions on NASA K9 Rover.

Compositional verification methods have also been studied on the Magic model extractor to verify various safety properties of about 74,000 lines of C code. These methods achieved two orders of magnitude space reduction compared to Magic's non-compositional analysis [12].

Compositional Verification Required Performance Level

Demonstration that the level of performance of the compositional verification methodology and tools are acceptable first requires the selection of a compelling example and to use it to quantify the scalability of the approach in verifying and validating complex aerospace systems. Suitable examples could include the ADAPT testbed (Power System for Avionics) developed by the IVHM teams at NASA Ames and the DAME framework (Drill Robotic platform) also developed at Ames. Both projects feature fault management systems integrating several types of fault diagnosis engines. Moreover, their modular approach to software health management illustrates the need for a compositional framework, which provides assurance that the composition of the solutions to smaller problems satisfies key system properties.

However, the combination of models often results in an explosion in the number of states that need to be explored during the verification process. This is known as the state space explosion problem. The first look at compositional verification performance will consist of evaluating the size (in number of states) of a behavioral model for a complex system and show that it can be efficiently reduced using the method of assume-guarantee reasoning mentioned above while maintaining the full diagnostic range of the complete model. Demonstrating that the method can effectively reduce the number of states in a complex (i.e., otherwise unverifiable) model is an important metric that can be used to characterize the scalability of the method. Other metrics might be the degree of interrelationships between models and the range of environmental (input) variables that can be handled. Ultimately, it must be demonstrated that a compositional framework can be created for a complex IVHM model that allows formal tools (e.g., model checkers) to be effectively used to verify the complete system in a relatively short amount of time. Hence, gains in terms of reducing state space size and analysis time are important metrics for assessing improvement in verification performance. Additionally, heuristic evaluation of how hard it is to generate environment assumptions (manually vs. automatically) will also be important to assess, as well as to characterize what properties can be addressed with the compositional framework.

Software Health Management

Software Health Management is a new research area aimed at developing tools and techniques that enable the detection, diagnosis, prognosis, and mitigation of errors and related adverse events caused or contributed to by software systems in aircraft. While software health management bears many similarities to health management of physical systems, there are important differences that must be taken into consideration. The most important consideration is that all software faults are design errors -- software does not fail or degrade in the physical sense. However, because aircraft software is inherently coupled with physical systems, many faults in aircraft software are triggered by interactions with unanticipated physical phenomena such as failed or degraded sensors, unforeseen system modes, unexpected user operation, and erroneous environmental assumptions. Thus, software health can only be assessed in the context of the larger system in which the software is embedded. Unfortunately, there is little reliable data concerning software failure. Specifically in [13] p. 39:

> *The lack of systematic reporting of significant software failures is a serious problem that hinders evaluation of the risks and costs of software failure and measurement of the effectiveness of new policies or interventions.*

This suggests an inherent difficulty in addressing detection and diagnosis of software faults. Furthermore, Avizienis, et al [14] observes that certain software faults are "*recognized as faults only after [...] a failure has ensued.*" It is possible that the first indication of a software fault is catastrophic system failure. A canonical example is the first flight failure of the Ariane 5 [15], which stemmed from sequence of seemingly reasonable design decisions. This highlights another observation of [13] p. 40 that "*by*

far the largest class of problems arises from errors made in the eliciting, recording, and analysis of requirements."

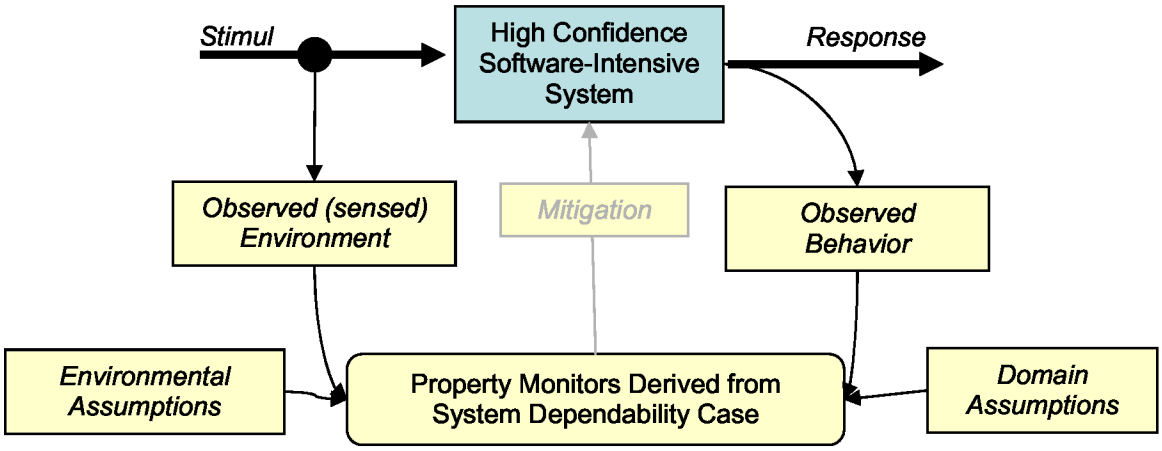

Figure 2: Notional Framework for Software Health Management

A central recommendation of the National Academies [13] is that dependable software systems should be developed with explicit claims and evidence to substantiate those claims, augmented with expertise in developing that class of systems. In light of these recommendations, research will be focused on developing a framework for (software) health management that involves:

- Explicit claims of system (and subsystem) requirements including assumptions about the application domain and environment in which the system is to operate;
- Evidence that software satisfies these explicit claims under the stated domain assumptions;
- Architectural principles, enforced by hardware mechanisms, that ensure that software behavior dependencies are traceable; and
- Mechanisms for correctly composing software systems from trusted components within the constraints imposed by the architectural principles.

To realize this framework, IVHM is exploring software health management in the context of system level dependability cases (Figure 2). Dependability cases are a mechanism recommended by Jackson et al [13] for managing the explicit claims and evidence in support of system dependability claims. Dependability cases are derived from safety cases [16], with a safety case broadly defined as [17]:

> *"A documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment" that should*
>
> - *make an explicit set of claims about the system*
> - *provide a systematic structure for marshalling the evidence*
> - *provide a set of safety arguments that link the claims to the evidence*

- *make clear the assumptions and judgments underlying the arguments*
- *provide for different viewpoints and levels of detail*

Bishop [17] further states that a safety case should consist of the following elements: a *claim* about a property of the system or some subsystem; *evidence* which is used as the basis of the safety argument; an *argument* linking the evidence to the claim, and an *inference* mechanism that provides the transformational rules for the argument.

A definition of dependability may be found in [18]:

> *Dependability of a computing system is the ability to deliver service that can justifiably be trusted. The service delivered by a system is its behavior as it is perceived by its user(s); a user is another system (physical, human) that interacts with the former at the service interface.*

A dependability case is formed from the evidence presented to justify an explicit dependability claim that makes clear which properties in the real world the system is intended to establish [13], and is the mechanism recommended for managing the explicit claims and evidence in support of system dependability claims summarized as:

> *Explicit claims – properties to be exhibited, assumptions about the environment upon which the claims are contingent, and the level of dependability claimed,*
>
> *Evidence – substantiate the dependability claim in the form of a dependability case arguing that the required properties follow from the combination of the properties of the system itself (the implementation) along with the environmental assumptions, typically containing results from testing, analysis, and development processes*

The central idea behind our approach to software health management is that any observed (sub) system behavior that is inconsistent with any explicit (sub) claim in a dependability case is evidence that either the system or its associated dependability case is flawed. In either case, there is reason to doubt the dependability of the system. Initial tasks will focus on detection and mitigation techniques, with the anticipation that more robust detection capabilities will lay a foundation for future investigations into diagnosis and prognosis.

State-of-the-Art in Software Health Management

There are examples in the literature that can be considered software health management techniques. Sha [19] outlines an architectural mitigation strategy based on run-time monitors coupled with simple, safe, but otherwise sub-optimal alternative solutions. The central idea is that it is easier to establish the safety of the monitors and the simple solution than the complex primary system. Monitoring of system requirements specifications during program execution is described in [20, 21]. Goldberg [22] advocates adapting ARINC 653 Health Monitoring mechanisms to support monitoring of software using formal models of expected behavior. Castelli et al [23] documents a proactive approach to a class of aging software faults. In this context, aging refers to run-

time degradation of software integrity due to resource exhaustion, data corruption, or accumulation of numerical errors. The strategy outlined is centered on periodically refreshing or restoring the state to eliminate the deleterious effects. This is a reasonable strategy for systems with long periods of continuous operation, and is worth considering for ground systems. However, for aircraft software systems, the flight duration is rarely more than ten hours. Airborne systems are already periodically restored to a known good state prior to every flight.

However, airborne systems may suffer from another form of software aging. Parnas [24] suggests two contributing factors for software aging: (1) not modifying software in response to evolving needs, and (2) modifying software in response to evolving needs. While there are mechanisms in place to manage changes to fielded software systems, this is an area with potential for either introducing new or unmasking existing software defects.

Baseline Datasets for Software Health Management

Another objective is to gain a better understanding of relevant software failure mechanisms for aircraft systems. There exist taxonomies of faults [14, 25]. We will determine which classification scheme is appropriate for aircraft systems. Nikora [26] is currently analyzing historical software fault data (using the classification suggested in [25]) from several robotic space exploration missions. There are currently several studies underway exploring various concepts and strategies for software health management. Unfortunately, the initial reports from these efforts funded by the IVHM project were not available in time for this baseline assessment.

Software Health Management Relevance to Aviation Safety

Although aviation software is subject to rigorous verification and validation, software-related issues have been implicated in some accidents and near misses [13]. Furthermore, aerospace software size and system complexity are expected to dramatically increase, as noted in [27]: "*Estimates on source lines of code for systems beyond the current generation of developing systems are several orders of magnitude higher – and will likely exceed one billion lines of code*". This increase in software size and system complexity will undoubtedly include the fielding of systems incorporating new software-based technologies for detection, diagnosis, prognosis, and mitigation of adverse events. Software health management seeks to address software-related issues by identifying inconsistencies between observed behavior and the explicit claims in the dependability case.

Of course, this also highlights a need to manage the interplay between the compositional verification strategies and software health management. The central strategy for compositional verification is assume-guarantee reasoning. The software health management philosophy is centered on the systems response to violations of assumptions. This leads to the question of how to manage compositional verification in the presence of assumption violations. Rushby [28] has previously explored this issue. He outlines a collection of key properties necessary to support modular certification. In addition to assume-guarantee reasoning, he highlights the necessity of an architecturally enforced

partitioning mechanism to ensure that composed components can only interact through their defined interfaces (even in the presence of some failure mechanism). In addition, he recognizes that assume-guarantee reasoning introduces circular dependencies that might lead to cascading failures, should some of the component assumptions be violated. His report outlines some strategies for resolving this difficulty, but additional investigations are warranted.

Hybrid Systems Analysis

Model-based Integrated Vehicle Health Management (IVHM) systems perform fault detection and diagnosis by monitoring a physical system and matching observations with a model of that system. In traditional state-transition models, states of a system change in discrete time steps. In continuous dynamical systems, states evolve continuously. Hybrid systems combine these two notions. They include both real-valued and discrete state variables, and their description involves a combination of differential equations and discrete state transitions. The hybrid systems formalism is a framework for modeling a large class of systems in which digital components or controllers interact with physical devices. Validation of models and verification of algorithms using formal, analytical techniques provide the rigor needed for dependable operation in critical subsystems.

This section heavily draws from material by SRI International, led by principal investigator John Rushby under a NASA cooperative agreement. It focuses primarily on the specific technique of hybrid abstraction. NASA's IVHM project intends to explore other aspects of hybrid systems analysis in the future.

Hybrid Systems Analysis State-of-the-art

Formal verification of hybrid systems poses theoretical and practical challenges. It is known that checking reachability properties for very simple classes of hybrid systems is undecidable. Even when semi-decision algorithms exist, hybrid models of real systems are too large and too complicated to be analyzed with such techniques. The most successful approaches to automate the verification of hybrid systems rely on abstraction.

In general, the goal of abstraction is to transform a complex system into a reduced form that is accessible to simpler analysis tools yet retains sufficient detail to establish relevant properties. Hybrid Abstraction is a new static analysis method [29 - 31] developed by Ashish Tiwari of SRI International that allows the principled construction of sound, finite-state abstractions of hybrid systems, while preserving sufficient properties for analysis (in the context of this paper we use "hybrid abstraction" to refer to this specific SRI technology). In Tiwari's framework, continuous dynamics are expressed using polynomial expressions, that is, the derivative with respect to time of a continuous state variable x is a polynomial function f of the state variables, where f may be nonlinear. The method derives a series of simplified expressions from f to use in the formulation of system properties.

SRI has developed a prototype mechanization of hybrid abstraction that is implemented in the open source tool called Hybrid SAL, which in turn is an extension of the model

checking tool SAL [32]. Given this preliminary implementation, it is possible, on a limited scale, to design and validate models used in model-based diagnosis. A potential limitation, however, is deduction involving nonlinear arithmetic. Ongoing improvements in this area and other features of the Hybrid SAL automation are needed to reduce hybrid abstraction to practice.

To assess the potential for hybrid analysis methods, a small modeling study was performed by NASA postdoctoral researcher Arturo Tejada [33]. The goal was to examine an emerging fault detection technology based on distributed Bragg fiber-optic strain sensors. Such sensors can be embedded in aircraft wings to continuously monitor surface strain during flight. Strain information and spectral analysis techniques can then be used to detect faults due to changes in the wing's physical parameters or to detect the presence of incipient cracks.

Working from the physical principles behind the modeling of vibrating structures such as cantilever beams (the natural model of a wing), Tejada reviewed two different classes of fault detection techniques and proposed a particular detection method for cracks in wings. The model requires an ability to handle fourth-order partial differential equations that are beyond the reach of current state-of-the-art analysis methods for hybrid system verification. The new work underway in the IVHM project will address this current limitation.

Hybrid Systems Adverse Events

Adverse events in the class of hybrid systems described here generally take the form of logical or mathematical flaws in models, algorithms and software implementations. A flaw in a system model or software component is a design defect rather than a degradation fault or a physical-failure fault. While software faults usually are introduced early in the life cycle, a latent fault might not be manifested until the component is operational. Conditions encountered during operation determine whether faulty software will be invoked, and whether its effects are immediate or delayed.

Current industry practice for managing software faults is based on the use of two broad categories of techniques. The first can be called static analysis methods, which include manual reviews and semi-automated checking of requirements, designs and software code. The methods are termed static analysis because the code is not executed. This is in contrast to the second category, dynamic analysis methods, which make extensive use of testing and simulation by executing the code which may be instrumented for monitoring and analysis. Techniques based on formal methods, of which hybrid abstraction is an instance, generally fall into the static analysis category. Research and development results are starting to make their way into industrial practice, bringing considerably more mathematical rigor into verification practices.

Hybrid Systems Analysis Baseline Datasets

Hybrid system modeling and analysis techniques have been pursued actively by researchers since the mid 1990s. Owing to the inherent difficulty of this domain, progress has been slower than that of other types of formal and analytical methods. One limitation

of the field is the intractability of automated deduction on the domain of nonlinear real arithmetic.

Reasoning over nonlinear inequalities is key to effective analysis of hybrid systems. Existing methods suffer from incompleteness or poor performance. Constructing the transition relation of a model involves proving or disproving formulas that are Boolean combinations of polynomial equalities and inequalities. The full first-order theory of the reals is decidable but it has a double exponential lower bound. The classical decision procedure for this theory is Collins's cylindrical algebraic decomposition (CAD) algorithm [34], but it is very complex and computationally expensive. Available implementations of this decision procedure can solve only very small problems.

SRI and its collaborators have developed a new algorithm called RAHD (Real Algebra in High Dimensions) that promises to be sound, complete and effective. It combines many existing algorithms so subproblems are solved using the cheapest technique. A heavily modified form of the CAD algorithm is a key element of RAHD, and the existing CAD implementation QEPCAD-B [35] is also incorporated as a back-end solver. SRI has created a prototype implementation of the RAHD algorithm, which is now undergoing test, evaluation, and refinement. After it is integrated with Hybrid SAL, the overall effectiveness of the hybrid abstraction technique will be enhanced.

Hybrid Systems Analysis Relationship to Aviation Safety

Hybrid systems analysis will be vital in the application of compositional verification techniques for verification of complex and modular IVHM systems. The analytical methods and tools under development will provide effective means to assess the correctness and safety of complex IVHM and fault-tolerant systems in vehicles that will operate in the Next Generation Air Transportation System. These analytical techniques also will contribute to increased confidence in the correctness of vehicle software, thereby enabling increased air-traffic efficiency while maintaining safety.

By taking the theoretical developments of hybrid abstraction and integrating them with a set of existing, SRI-maintained tools, this new analytical method will become available to all segments of the aviation community. Engineers who choose to create hybrid models can make use of this capability to verify system properties rigorously and do so at an early stage in the life cycle.

Hybrid Systems Analysis Performance Required

As with many analytical methods, verification of hybrid systems lacks established performance requirements. Moreover, the hybrid abstraction technique is too new to have generated any significant performance data. Given that implementations of these techniques typically are used in the context of modeling or engineering tools, acceptable performance often is dictated by the nature and use of the tools. Interactive tools, for example, require response times in seconds or minutes to be acceptable to a human user. Batch-oriented tools, however, might have the luxury of longer execution times that can run into hours.

For hybrid systems, one crude measure of performance is occasionally mentioned, namely, the number of continuous (real-valued) variables that the analysis methods can handle. When a hybrid system model with more than this number of continuous variables is attempted, the execution time quickly becomes excessive. In the case of hybrid system techniques based on the use of reachability analysis, the limit is typically five or six continuous variables. In contrast, the hybrid abstraction technique has shown tractability when the models contain 15 or more continuous variables. With ongoing developments in decision procedures for real arithmetic, such as RAHD, the outlook is for even more improvement.

Prioritization

There are numerous verification, validation, and certification challenges that must be addressed in the deployment of advanced IVHM technologies. These challenges are not unique to IVHM. Adaptive controls [36, 37], uninhabited autonomous air vehicles [38], and development of the next generation air transportation system [39] are some examples of emerging systems that will necessitate increased levels of automation that rely upon advanced technologies. We propose prioritizing candidate enabling verification and validation capabilities based upon relevance to system-level dependability cases, relevance to the specific needs of IVHM technology, and relevance to broader aviation safety verification and validation needs. These considerations are outlined in table 2.

Considerations for Prioritizing Integrity Assurance Research	
Criteria	*Notes*
Relevance to IVHM	
Specific IVHM advanced information processing technology under consideration	Examples include model-based diagnosis, feature extraction and pattern analysis, etc.
Specific IVHM systems that employ embedded computation integrated with physical processes.	Examples include health state monitoring, damage mitigating control, condition-based maintenance
Relevance to system-level dependability cases	
Alignment of technology with identifying inconsistencies between observed behavior and the explicit claims in the dependability case	Examples include instrumentation concepts, runtime monitoring, and architectural enforcement mechanisms.
Relevance to broader aviation safety V&V	
Safe composition of rigorously verified systems and sub-systems	Tools, techniques, and methods that are applicable to verification, validation, and certification of complex dynamic systems
Existing air traffic and vehicle health management	Scalable techniques applicable to increasing levels of automation and decreasing ability of flight and ground crew to provide safety margins
Developing air traffic and vehicle health management	
Future generation air traffic and vehicle health management	
Alignment of technology development with national and international committees, standard-setting organizations, and technical working groups	Examples include RTCA, SAE, and recommendations from the National Academies and the President's Council of Advisors on Science and Technology

Table 2: Possible prioritization criteria for enabling capabilities

The introduction of software health management as a research topic suggests a viewpoint for prioritizing enabling V&V research, and reinforces the relevance of the current research activities underway within the IVHM project. The *software health management* effort of the IVHM project specifically seeks to define a framework for identifying inconsistencies between observed behavior and the explicit claims in the dependability case. As noted earlier, there currently exist examples of technologies that may be considered software health management techniques. But unique to NASA's IVHM approach is the provision of a framework for proving system safety in the context of a system that is instrumented in such a way as to examine whether inconsistencies exist between the explicit claims and assumptions under which the system design was deemed safe and the actual behavior of the fielded system. Once a framework for observing inconsistencies is established, further investigation into diagnosis, prognosis, and mitigation approaches will be possible. Even though NASA's IVHM research into software health management has only just begun, there are clearly areas of research that will be enabling to this framework. Areas currently being pursued in NASA's IVHM project include the composing of components that certifiably satisfies global requirements [40], rigorous checking of system requirements at runtime [20], compositional verification and assume-guarantee [7], and tools for verification of system models that are comprised of both discrete and continuous variables [29]. While these enabling capabilities provide a solid basis to support the software health management framework, inevitably other research opportunities will be presented as the software health management work proceeds. For example, research into new architecture principles is needed that will ensure that observed inconsistencies are due exclusively to either faults in the software design or flaws in the safety argument. These principles would be shaped from architectural requirements derived from emerging software health management frameworks.

To effect a comprehensive estimation of vehicle-level health state, integrated health management necessitates that there be an exchange of data and information among distributed systems, subsystems, and components that are highly networked. Because IVHM systems will be introduced at every level of this hierarchy, and will likely share common computing resources within an Integrated Modular Avionics (IMA) architecture framework, there is a pressing need to address verification and validation capabilities that support modular certification concepts [28]. NASA's IVHM work in compositional verification and architectural enforcement are supportive of these modular certification concepts.

IVHM systems are comprised of embedded computers that incorporate networks of sensor modules for acquiring physical parameters and networks of actuator modules for control, optimization, and mitigation. These are, in effect, next generation cyber-physical systems (CPS) [41] that will require new tools and methods for verification and validation of the complex interactions between modules [27]. IVHM is conducting research into developing *hybrid abstraction* tools that will support the analysis of systems that exhibit both discrete and continuous behavior as is typically found in the cyber-physical realm.

Recognizing that the V&V challenges associated with deploying systems incorporating advanced technology is not unique to IVHM, relevance to broader aviation safety verification and validation needs should also be a factor when prioritizing research focus areas. Zemrowski [42] examines the engineering challenges faced by NextGen in order to maintain safety while increasing capacity, noting that

> *"Reliability and availability will be even more important than they are in today's most critical applications. Because of the safety criticality, special software design techniques may be required to be able to use certified operating systems and be able to partition software so that uncertified software does not adversely affect critical routines".*

Furthermore, relevance of NASA's IVHM verification and validation research initiatives to national and international standard-setting organizations and technical working groups is also an important consideration for portfolio prioritization. The IVHM project's Technical Plan [1] calls out the need for coordination across the broader research community, and the coordination with standard-setting bodies and technical working groups is particularly significant to verification and validation as many requirements will be driven by the system of rules and regulations that govern the use of advanced technology in the National Airspace System.

Conclusion

NASA's Integrated Vehicle Health Management project is conducting research aimed at enabling capabilities to support the verification and validation of IVHM technologies. While these enabling capabilities are necessary for fielding IVHM systems, they are certainly not unique to IVHM. Technologies that involve dynamic and non-deterministic algorithms for pattern recognition, feature extraction, and fusion (to name a few) are germane to IVHM as well numerous emerging applications such as adaptive flight controls, autonomous air vehicles, and next generation air transportation systems. Currently NASA's IVHM project is focusing on *compositional verification* for providing strong guarantees that the verification of a system or sub-system can take advantage of the verification of individual components, *hybrid systems analysis* for providing analysis capability of systems comprised of both continuous and discrete behavior, and a new IVHM concept referred to as *software health management* which seeks to address software-related faults in the context of system-level dependability cases. A key consideration in software health management is the employment of *architecturally enforced partitioning mechanisms* to ensure that system and subsystem faults are contained in a manner that provides unambiguous traceability to faulty software behavior. Another consideration is the management of compositional verification, which is centered on assume-guarantee reasoning, in the presence of violations of these assumptions, which is the central philosophy of software health management. The selection and priority of these focus areas is based upon relevance to the specific needs of the IVHM project, impact on software health management, and applicability to broader aviation safety needs. Software health management is a new research endeavor that will undoubtedly introduce new research needs as the work progresses.

References

1. "Integrated Vehicle Health Management Technical Plan, Version 2.01," National Aeronautics and Space Administration, August 14, 2008 <http://www.aeronautics.nasa.gov/nra_pdf/ivhm_tech_plan_c1.pdf>
2. Lichtenwalner, P.F., White, E.V., and Baumann, E.W., "Information processing for aerospace structural health monitoring," SPIE Vol. 3326, Pg. 406-417 1998.
3. RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," December 1992
4. Jacklin, S., Schumann, J., Gupta, P., Lowry, M., Bosworth, J., Zavala, E., Hayhurst, K., Belcastro, Celeste, and Belcastro, Christine, "Verification, Validation, and Certification Challenges for Adaptive Flight-Critical Control System Software," AIAA-2004-5258, AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, Rhode Island, Aug. 16-19, 2004
5. Jones, C. B., "Tentative steps toward a development method for interfering programs." ACM Trans. on Prog. Lang. and Sys., 5(4):596–619, Oct. 1983.
6. Pnueli, A., "In transition from global to modular temporal reasoning about programs," In Logics and models of concurrent systems, pages 123–144, 1985.
7. Giannakopoulou, D., Pasareanu, C., and Cobleigh, J.M., "Assume-guarantee Verification of Source Code with Design-Level Assumptions," ICSE'2004.
8. Pasareanu, C.S., Giannakopoulou, D., Gheorghiu, M., Cobleigh, J.M., and Barringer, H. "Learning to Divide and Conquer: Applying the L* Algorithm to Automate Assume-Guarantee Reasoning". Springer Journal of Formal methods in System Design, special issue on Compositional Reasoning Volume 32, Number 3, June 2008.
9. Giannakopoulou, D., Pasareanu, C., and Barringer, H. "Assumption Generation for Software Component Verification", in Proc. of the 17th IEEE International Conference on Automated Software Engineering (ASE 2002). September 2002, Edinburgh, UK. (ACM distinguished paper award)
10. Cobleigh, J.M., Giannakopoulou, D., and Pasareanu, C.S. "Learning Assumptions for Compositional Verification", in Proc. of the 9th International Conference for the Construction and Analysis of Systems (TACAS 2003). April 2003, Warsaw, Poland. Springer, LNCS 2619.
11. Giannakopoulou, D., and Pasareanu, C.S. "Interface Generation and Compositional Verification in JavaPathfinder", in Proceedings of FASE 2009, York, UK, March 2009
12. Chaki, S., Clarke, E., Giannakopoulou, D., and Pasareanu, C., "Abstraction and assume-guarantee reasoning for automated software verification," RIACS TR 05.02, October 2004.
13. Jackson, D., Thomas, M., and Millett, L. I., Eds. "Software for Dependable Systems: Sufficient Evidence?" National Academies Press, May 2007.
14. Avizienis, A., Laprie, J., Randell, B., and Landwehr, C., "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE Transactions On Dependable and Secure Computing, Vol. 1, No. 1, pp. 11—33, January-March 2004.
15. Lions, et al., "Ariane 5 Flight 501 Failure, Report by the Inquiry Board," July 1996. (retrieved from http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html

16. Maxion, R.A., and Olszewski, R.T., "Improving software robustness with dependability cases," Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on 23-25 June 1998 Page(s):346 – 355
17. Bishop, P.G., and Bloomfield, Robin E., ''The SHIP Safety Case Approach,'' SafeComp95, Belgirate, Italy 11-13 October 1995, pp 437-451, published by Springer (ed. Gerd Rabe)
18. Avizienis, A., Laprie, J., and Randell, B., "Fundamental Concepts of Computer System Dependability," IARP/IEEE-RAS Workshop on Robot Dependability: Technological Challenge of Dependable Robots in Human Environments, Seoul, Korea, May 21-22, 2001
19. Sha, L., "Using Simplicity to Control Complexity," IEEE Software, July/August 2001.
20. Chen, F., and Rosu, G., "MOP: An efficient and generic runtime verification framework," Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications 2007, pp 569 – 588
21. Feather, M. S., Fickas, S., van Lamsweerde, A., and Ponsard, C., "Reconciling System Requirements and Runtime Behavior," Proceedings of the 9th International Workshop on Software Specification and Design, April 1998
22. Goldberg, A., and Horvath, G., "Software Fault Protection with ARINC 653," IEEE Aerospace Conference, March 2007.
23. Castelli, V., Harper, R.E., Heidelberger, P., Hunter, S. W., Trivedi, K. S., Vaidyanathan, K., and Zeggert, W. P., "Proactive Management of Software Aging," IBM J. Res. & Dev., Vol. 45, No. 2, March 2001.
24. Parnas, D. L., "Software Aging," Proceedings of the 16th International Conference on Software Engineering, pp. 279—287, 1994.
25. Grottke, M., and Trivedi, K., "Fighting Bugs: Remove, Retry, Replicate, and Rejuvenate," IEEE Computer, pp. 107 – 109, February 2007.
26. Nikora, A., "Classifying Software Faults to Improve Fault Detection Effectiveness," NASA OSMA Software Assurance Symposium, September 2007. (retrieved from http://sarpresults.ivv.nasa.gov/ViewResearch/130.jsp)
27. Winter, D. C., Statement of Boeing's Perspective on Cyber-Physical Systems Research, Hearing on Networking and Information Technology Research and Development Program, Committee on Science and Technology, U. S. House of Representatives, July 31, 2008 http://democrats.science.house.gov/Media/File/Commdocs/hearings/2008/Full/31july/Winter_Testimony.pdf
28. Rushby, J., "Modular Certification," NASA/CR-2002-212130 December 2002
29. Tiwari, A., and Khanna, G., "Series of abstractions for hybrid automata," In C. J. Tomlin and M. R. Greenstreet, editors, Hybrid Systems: Computation and Control HSCC, volume 2289 of LNCS, pages 465–478. Springer, March 2002.
30. Tiwari, A., "An algebraic approach for the unsatisfiability of nonlinear constraints," In L. Ong, editor, Computer Science Logic, 14th Annual Conf., CSL 2005, volume 3634 of LNCS, pages 248–262. Springer, August 2005.
31. Rodriguez-Carbonell, E., and Tiwari A., "Generating polynomial invariants for hybrid systems," In M. Morari and L. Thiele, editors, Hybrid Systems: Computation

and Control, HSCC 2005, volume 3414 of LNCS, pages 590–605. Springer, March 2005.
32. Leonardo de Moura, Owre S., Rueß H., Rushby J., Shankar N., Sorea M., and Tiwari A., "SAL 2," In Rajeev Alur and Doron Peled, editors, Computer-Aided Verification, CAV '2004, volume 3114 of Lecture Notes in Computer Science, pages 496–500, Boston, MA, July 2004. Springer-Verlag. SAL home page: http://sal.csl.sri.com/.
33. Tejada, A., "A Mode-Shape-Based Fault Detection Methodology for Cantilever Beams," NASA/CR-2009-215721, May 2009.
34. Collins, G. E., "Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition," In Proceedings Second GI Conference on Automata Theory and Formal Languages, volume 33 of Lecture Notes in Computer Science, pages 134–183. Springer-Verlag, 1975.
35. Hong, H., Quantifier elimination in elementary algebra and geometry by partial cylindrical algebraic decomposition (version 13). http://www.gwgd.de/~cais/systeme/scalib, 1995.
36. Rushby, J., "How Do We Certify for the Unexpected?" AIAA 2008-6799, AIAA Guidance, Navigation, and Control Conference August 2008, Honolulu, Hawaii
37. Jacklin, S. A., "Closing the Certification Gaps in Adaptive Flight Control Software," AIAA Guidance, Navigation, and Control Conference, Honolulu, HI, August 2008
38. Halski, D., Barhorst, J., Swearingen, K., and Urnes, J., "Software V&V Challenges for Uninhabited Autonomous Air Vehicles," AIAA-2004-5255, AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, Rhode Island, Aug. 16-19, 2004
39. OIG Statement CC-2008-118, "Status of FAA's Efforts To Develop the Next Generation Air Transportation System," September 11, 2008 <http://www.oig.dot.gov>
40. Manolios, P., "Automating Component-Based System Assembly," AIAA 2008-6803 AIAA Guidance, Navigation and Control Conference and Exhibit 18 - 21 August 2008, Honolulu, Hawaii
41. Lee, E. A., "Cyber Physical Systems: Design Challenges," 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing, May 2008
42. Zemrowski, K. M., "Impacts of Increasing Reliance on Automation in Air Traffic Control Systems," SysCon 2008 – IEEE International Systems Conference, Montreal, CA, April 7-10, 2008

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-06-2009	Technical Memorandum	

4. TITLE AND SUBTITLE
Baseline Assessment and Prioritization Framework for IVHM Integrity Assurance Enabling Capabilities

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Cooper, Eric G.; Di Vito, Benedetto L.; Jacklin, Stephen A.; Miner, Paul S.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
645846.02.07.07.15.02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L-19688

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/TM-2009-215764

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 62
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Fundamental to vehicle health management is the deployment of systems incorporating advanced technologies for predicting and detecting anomalous conditions in highly complex and integrated environments. Integrated structural integrity health monitoring, statistical algorithms for detection, estimation, prediction, and fusion, and diagnosis supporting adaptive control are examples of advanced technologies that present considerable verification and validation challenges. These systems necessitate interactions between physical and software-based systems that are highly networked with sensing and actuation subsystems, and incorporate technologies that are, in many respects, different from those employed in civil aviation today. A formidable barrier to deploying these advanced technologies in civil aviation is the lack of enabling verification and validation tools, methods, and technologies. The development of new verification and validation capabilities will not only enable the fielding of advanced vehicle health management systems, but will also provide new assurance capabilities for verification and validation of current generation aviation software which has been implicated in anomalous in-flight behavior. This paper describes the research focused on enabling capabilities for verification and validation underway within NASA's Integrated Vehicle Health Management project, discusses the state of the art of these capabilities, and includes a framework for prioritizing activities.

15. SUBJECT TERMS
Software health management; Compositional verification; Hybrid systems analysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	22	(443) 757-5802